Tampereen ammattikorkeakoulu

Bronto Duties -ohjelman kolmiulotteisen näkymän toteutus Three.js-kirjastolla

Juho Sinisalo

OPINNÄYTETYÖ
Toukokuu 2020

Tietojenkäsittely
Ohjelmistotuotanto

TIIVISTELMÄ

Tampereen ammattikorkeakoulu
Tietojenkäsittely
Ohjelmistotuotanto

SINISALO, JUHO:
Bronto Duties -ohjelman kolmiulotteisen näkymän toteutus Three.js-kirjastolla

Opinnäytetyö 32 sivua
Toukokuu 2020

Opinnäytetyössä kuvaillaan Bronto Skyliftin tilaaman Bronto Duties -ohjelman kolmiulotteisen näkymän eri osia ja rakennetta. Työn tavoitteena oli tehdä Bronto Skyliftin vaatimusten mukainen kolmiulotteinen näkymä nostolavojen ulottumien etäisyyksien havainnointiin. Työssä tuotettiin näkymä käyttäen nykyaikaisia selainpohjaisia teknologioita kuten Reactia, Reduxia ja Three.js:ää.

Kolmiulotteinen näkymä on rakenteeltaan päällepäin yksinkertainen, mutta sisäisesti monimutkainen. Rakenteen yksinkertaisuuden takia merkittävä osa näkymää hallinnoivasta logiikasta on kasautunut yhteen tiedostoon, mikä hankaloittaa ohjelman luettavuutta. Näkymää hallinnoiva logiikka rakentaa ohjelman käyttäjän syötteiden perusteella lukuisia eri näkymän osia, jotka Three.js-kirjasto piirtää käyttäjän havainnoitavaksi näkymäksi. Näkymän eri osiin kuuluvat esimerkiksi nostolava-ajoneuvo, ulottumakäyrä, rakennus ja mitta-asteikko.

Opinnäytetyön tuloksena tehtiin sisäisesti monimutkainen, mutta tavoitteet saavuttava kolmiulotteinen näkymä, josta on hyötyä Bronto Skyliftille. Näkymän lopullisiin ominaisuuksiin kuuluu muun muassa kaksi kameraesitystä, mahdollisuus tutkia nostolavan ulottumakäyrää usealla eri asetuksella, säädettävän rakennuksen ja tuulimyllyn esitys sekä mahdollisuus tutkia ulottumakäyrän ja rakennuksen välistä leikkausta.

Asiasanat: kolmiulotteinen näkymä, three.js, nostolava

ABSTRACT

Tampereen ammattikorkeakoulu
Tampere University of Applied Sciences
Degree Programme in Business Information Systems
Software Development

SINISALO, JUHO:
Implementation of a Three-dimensional View with Three.js Library in Bronto Duties

Bachelor's thesis 32 pages
May 2020

This thesis describes the implementation of a three-dimensional view in the Bronto Duties program. The aim of this thesis was to deliver a three-dimensional view for the purpose of viewing the reach and other qualities of an aerial platform as described by the client of the program, Bronto Skylift. The purpose of the thesis was to create the three-dimensional view using the latest web-based technologies, such as React, Redux and Three.js.

The three-dimensional view is comprised of multiple intertwined parts, which on the surface appear simple. However, internally they are complex and hard to handle. A significant portion of the logic that handles the three-dimensional view is all in the same file, which makes the program hard to develop for. During the program's lifetime, the handling logic creates multiple objects, according to the program's users input, which are then drawn on the three-dimensional view by the Three.js library. Objects drawn on the view include, but are not limited to, aerial platforms, outreaches, buildings and a scale.

The result of this thesis is a three-dimensional view that is according to the demands of the client, fulfilling the aim of the thesis. The final features of the program include, but are not limited to, two camera presentations, possibility of viewing multiple different outreach positions and settings, an adjustable building or windmill and a possibility to turn on intersections between the building and the outreach.

Key words: three-dimensional view, three.js, aerial platform

SISÄLLYS

1 JOHDANTO

Opinnäytetyön tavoitteena on Bronto Skyliftin vaatimusten mukainen kolmiulotteinen näkymä uudistettuun Bronto Duties -ohjelmaan. Bronto Duties -ohjelma oli Bronto Skyliftin valmistama ohjelma nostolavojen ulottumien etäisyyksien, korkeuksien ja muotojen havainnointiin. Ohjelma oli kuitenkin vanhentunut ja kaksiulotteinen, joten Bronto Skylift tilasi ohjelmasta uuden version.

Opinnäytetyön tarkoituksena on tuottaa Bronto Duties -ohjelman kolmiulotteinen näkymä käyttäen moderneja selainpohjaisia teknologioita kuten React, Redux, sekä kirjastoja kuten Three.js. Lopputuloksena tulisi olla tavoitteet ja vaatimukset täyttävä kolmiulotteinen näkymä.

Opinnäytetyössä kuvaillaan lopullisen kolmiulotteisen näkymän ominaisuuksia, rakennetta, osia, sekä työtä tehdessä ilmeneviä ongelmia ja mahdollisia ratkaisuja. Toisessa luvussa käydään läpi ohjelman taustaa, kolmannessa luvussa esitellään yleisesti näkymässä käytetyt teknologiat, neljännessä luvussa kuvaillaan perusteellisesti kolmiulotteisen näkymän vaatimuksia, rakennetta ja osia. Viimeisessä luvussa lyhyesti pohditaan työn lopputuloksia.

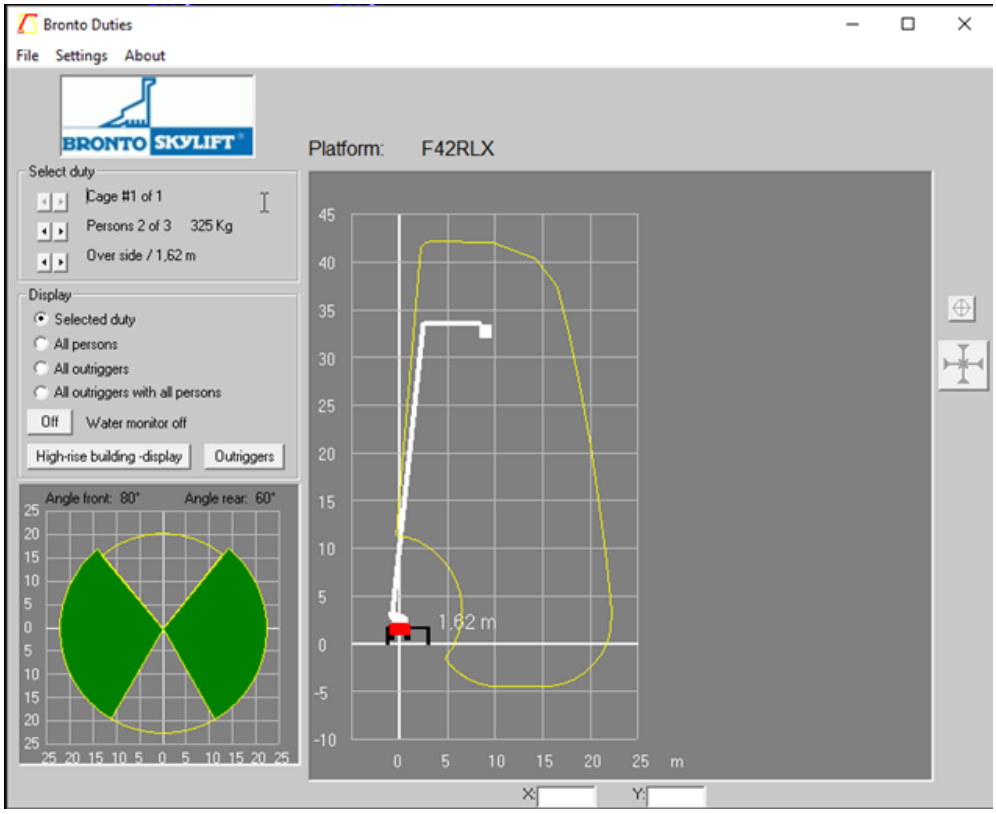
2 TAUSTATIEDOT

2.1 Bronto Skylift

Bronto Duties -projektin asiakkaana toimii suomalainen Bronto Skylift -yritys. Yritys valmistaa ja toimittaa maailmanlaajuisesti kuorma-autojen alustoille rakennettuja nostolava-ajoneuvoja (Bronto Skylift 2019).

2.2 Bronto Duties

Bronto Duties -ohjelma (kuva 1) oli alun perin Bronto Skyliftin kehittämä työpöytäsovellus, jolla pystyttiin havainnoimaan valmistettujen nostolavojen ulottumien etäisyyksiä, korkeuksia ja muotoja.



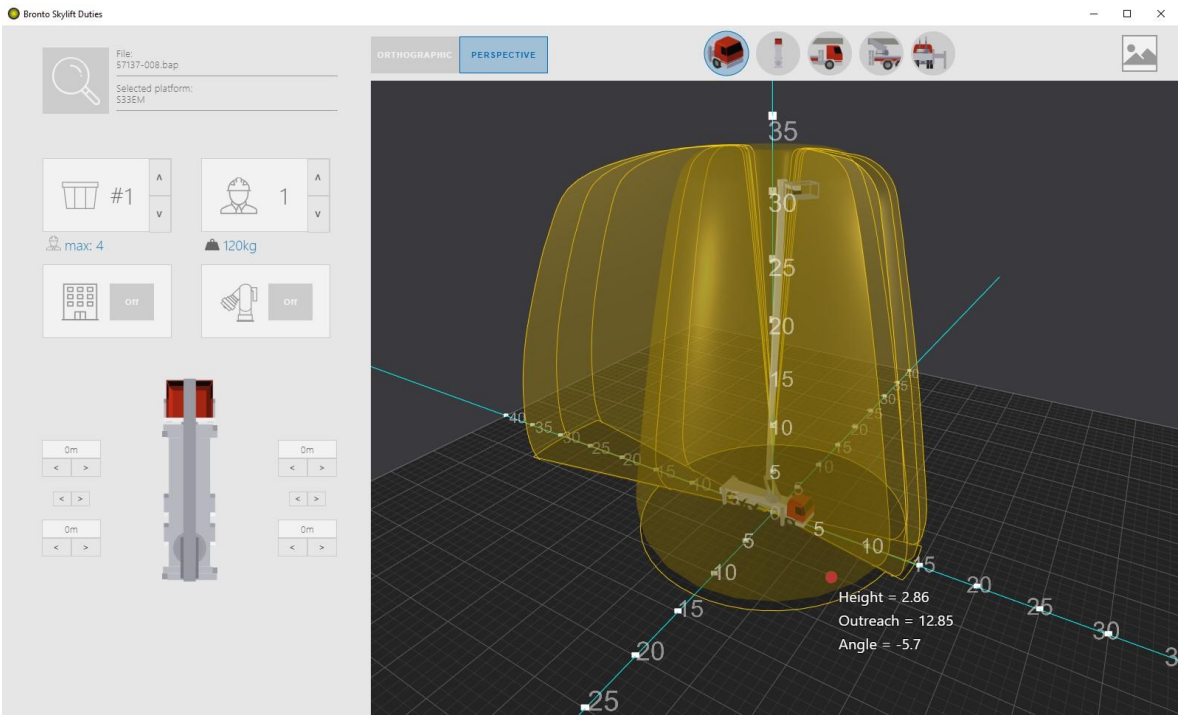
KUVA 1. Vanha Bronto Duties -ohjelman käyttöliittymä (Bronto Skylift 2019)

Sovelluksen toiminta oli yksinkertainen: ”valitse haluamasi nostolava-ajoneuvo ja tutki sovelluksen siitä piirtämää kaksiulotteista ulottumakäyrää”. Ulottumakäyrä

on käyrä, joka kuvailee pituutta, korkeutta sekä muotoa. Ulottumakäyrän pystyi piirtämään ajoneuvon näkökulmasta joko ylhäältä, edestä, takaa tai sivulta. Sovelluksella pystyi myös säätämään ajoneuvon tukijalkojen etäisyyksiä, korin mallia ja korissa olevien henkilöiden määrää, mitkä vaikuttivat ulottumakäyrän kokoon ja muotoon. Sovellus oli yksinkertainen, mutta vanha eikä enää toiminut kunnolla uudempien Windows-käyttöjärjestelmien kanssa. Bronto Skylift tilasi uuden ja päivitetyn version ohjelmasta vuonna 2019.

Uudessa versiossa päätoiminnallisuuden tulisi pysyä ennallaan, mutta kolmiulotteisena. Lisäominaisuuksiin kuului perinteinen kaksiulotteinen näkymä, rakennuksen kuvaus sekä mahdolliset leikkauspisteet (eng. intersection) rakennuksen ja ulottumakäyrän kanssa. Käyttöliittymältään ja ulkonäöltään ohjelman tuli olla moderni, hyvännäköinen ja helppokäyttöinen. Kuvassa 2 näkyy lopullinen versio uudesta käyttöliittymästä.

Uuden sovelluksen toteutuksessa päädyttiin web-tekniikoihin perustuviin viitekehyksiin ja kirjastoihin. Web-tekniikoilla rakennettu sovellus voitaisiin myöhemmässä vaiheessa vähällä vaivalla siirtää työpöydältä myös mobiililaitteisiin tai jopa pilveen.



KUVA 2. Uusi Bronto Duties -ohjelman käyttöliittymä

3 KÄYTETYT TEKNIIKAT

3.1 Yleistä

Teknisesti sovellus koostuu kolmesta osasta: käyttöliittymästä, kolmiulotteisesta näkymästä ja ajoneuvojen tiedostojen jäsentelystä.

Käyttöliittymä on rakennettu React-viitekehyksellä, kolmiulotteinen näkymä on tehty Three.js-nimisellä kirjastolla ja tiedostojen jäsentelyyn toteutettiin tiedostoihin mukautettu ratkaisu. Osien välinen kommunikaatio toimii Redux-kirjaston avulla.

3.2 React

React on ”JavaScript-kirjasto käyttöliittymien rakentamiseen” (React 2019). Reactin avulla monimutkaisten käyttöliittymien tekemisestä tulee yksinkertaista ja nopeaa. Oikein käytettynä React luonnollisesti jäsentelee sovelluksen käyttöliittymän eri osat omiin tiedostoihinsa ja luokkiinsa. Reactin avulla käyttöliittymän eri osat voivat vaivatta keskustella keskenään.

3.3 Redux

Redux on ”ennakoitava tilan säiliö JavaScript-applikaatioille” (Redux 2019). Reduxin avulla voi hallita sovelluksen tilaa (eng. state), jolloin sovelluksen eri osat voivat sulavasti vaihtaa tietoja ja laukaista (eng. trigger) eri komentoja. Redux tekee sovellukseen muistinvaraisen tietokannan, josta sovelluksen eri osat voivat tilata (eng. subscribe) haluamiaan tietoja.

Sovelluksessa Redux tulee käyttöön kaikkien osien välisenä tiedon välittäjänä. Kun käyttäjä säätää ajoneuvon tukijalkojen etäisyyttä, käyttöliittymä laukaisee Redux-komennon, joka vaihtaa tietokannassa olevan käyttöliittymän tilan uuteen tilaan. Tiedoston jäsentelystä vastaava sovelluksen osa huomaa tämän ja laittaa

tietokantaan käyttöliittymän tilaa vastaavan ajoneuvon tilan. Lopuksi kolmiulotteinen näkymä tekee uudesta ajoneuvon tilasta vastaavan ulottumakäyrän.

3.4 Three.js

Three.js on JavaScript-pohjainen kirjasto kolmiulotteisten mallien esittämiseen ja primitiivisten muotojen generointiin. Kolmiulotteinen näkymä on rakennettu täysin käyttäen Three.js-kirjastoa ja sen ominaisuuksia. Three.js-kirjasto on tehty mahdollisimman yleisesti käytettäväksi, joten tätä sovellusta varten joitain osia kirjaston lähdekoodista piti muokata.

4 KOLMIULOTTEINEN TOTEUTUS

4.1 Näkymän vaatimukset

Kolmiulotteinen näkymä on sovelluksen olennaisin ja monimutkaisin osa, joten sille oli projektin alussa asetettu muutamia vaatimuksia. Osa vaatimuksista on asioita, joita vanha sovellus sisälsi, mutta nyt ne pitäisi tehdä kolmiulotteisena.

Tärkein toiminnallisuus on kolmiulotteinen esitys käyttäjän valitsemasta ajoneuvon konfiguraatiosta. Näkymän tulisi näyttää staattinen esitys ajoneuvosta ja sen varsistosta, sekä varsiston suurin mahdollinen ulottuma kolmiulotteisena käyränä ajoneuvon ympärillä. Käyttäjän pitäisi myös pystyä kiertämään, liikuttamaan ja suurentamaan (eng. zoom) näkymän kameraa.

Vanhaan sovellukseen perustuen, uuden näkymän pitäisi myös tukea vanhan mallista ”kaksiulotteista” näkymää, eli käyttöliittymässä tulisi olla nappi, joka lukitsee kameran kuvakulman ja muuttaa kameran perspektiiviä niin, että kolmiulotteinen näkymä näyttäisi kaksiulotteiselta. Tätä ominaisuutta hyödynnettäisiin tutkittaessa käyrää sivuilta tai suoraa ylhäältäpäin katsottuna.

Täysin uutena ominaisuutena näkymän tulisi tukea käyttäjän itse määrittelemää rakennusta, sekä visuaalisesti esittää rakennuksen ja ulottumakäyrän välistä leikkausta (eng. intersection). Rakennus tulisi esiin käyttöliittymän nappia painamalla ja käyttäjä pystyisi antamaan rakennukselle mielivaltaisen koon, paikan ja kulman. Tämän jälkeen käyttäjä pystyisi näkemään rakennuksen ja ulottumakäyrän välisen leikkauksen.

Tyylillisesti näkymässä tulisi painottaa eri osien selkeyttä. Näkymässä tulisi olla jonkinlainen etäisyyksiä havainnoiva elementti kuten mitta-asteikko. Ulottumakäyrästä pitäisi käyttäjälle helposti tulla ilmi käyrän muoto ja suuruus. Rakennuksen ja käyrän leikkausta tutkittaessa leikkauksen tulisi olla pääosassa ja erittäin selkeästi näkyvissä.

4.2 Näkymän rakenne

Projektin alkuvaiheissa tehty yksinkertainen rakenne näkymälle ei juurikaan kehittynyt projektin edetessä, mikä aiheutti ongelmia projektin loppuvaiheessa. Näkymän kaikki logiikka kasautui yhteen osaan, eikä sitä enää nopeasti saanut uudelleen jäsenneltyä, joten näkymästä tuli sisäisesti vaikeasti luettava.

Näkymän rakentaminen alkaa CanvasEntryPoint-oliosta, josta määritellään käyttäjän käyttöliittymän asetusten perusteella oikean kokoinen. CanvasEntryPoint-olioon upotetaan rakentamisen jälkeen SceneManager-olio, joka puolestaan rakentaa Three.js-kirjaston näkymän.

SceneManager-olio vastaa näkymän rakentamisen jälkeen kaikesta käyttäjän antamasta syötteestä ja muokkaa näkymää syötteiden mukaan. Käyttäjän syötteen tullessa SceneManager-olio poistaa kaikki syötteeseen perustuvat osat ja rakentaa ne uudelleen uusien syötteiden perusteella.

SceneManager-olion rakentamat osat perustuvat SceneSubject-nimiseen yksinkertaiseen luokkaan. SceneSubject-oliot ovat osia, jotka Three.js piirtää näkymälle käyttäjän nähtäväksi. SceneSubject-olioita ovat näkymältä vaaditut osat kuten ajoneuvo, ajoneuvon varsisto, ulottumakäyrä, mutta myös yleisesti kolmiulotteiseen näkymään kuuluvat osat kuten näkymän valaistus ja kamera.

4.3 Näkymän osat

4.3.1 CanvasEntryPoint

CanvasEntryPoint-luokka on yksinkertainen osa, jonka tarkoitus on alustaa koko näkymä. Three.js-kirjasto tarvitsee toimiakseen HTML5-canvas-elementin, jonka CanvasEntryPoint-olio tekee oikean kokoiseksi ohjelman käynnistyessä, sekä ohjelman ikkunan kokoa muuttaessa. HTML5-canvas-elementin rakentamisen jälkeen CanvasEntryPoint-olio rakentaa SceneManager-olion.

CanvasEntryPoint-olio vastaa myös ohjelman sisäisestä päivityssilmukasta. Ohjelman käynnistyessä päivityssilmukka käynnistyy ja rajoittaa ohjelman virkistystaajuuden haluttuun taajuuteen. Tämän ohjelman kannalta silmukasta tuleva päivitys ei kuitenkaan ole olennainen, sillä mikään ohjelmassa ei tapahdu itsestään, eikä mikään näkymällä liiku ilman käyttäjän syötettä. Päivityksen tehtävänä on ainoastaan piirtää Three.js-kirjaston näkymä ja piirtää käyttäjän hiiren kohdalle pieni merkki ja havainnollistaa ulottumakäyrän arvot hiiren kohdalla, mikäli käyttäjän hiiri sattuu olemaan ulottumakäyrällä.

Päivitys myös kutsuu kaikkien näkymässä olevien SceneSubject-olioiden omaa päivityskutsua, mutta suurimmalla osalla olioista tämä kutsu on tyhjä.

4.3.2 SceneManager & SceneSubject

SceneManager-luokka on ohjelman keskeisin ja sisäisesti monimutkaisin osa. SceneManager-luokka vastaa ohjelmassa käytännössä kaikesta käyttäjän syötteestä ja nimensä mukaisesti hallitsee Three.js-kirjaston näkymää.

SceneSubject-luokka on yksinkertainen malli, johon kaikki SceneManager-luokan hallinnoimat oliot perustuvat. SceneSubject-luokka määrittelee päivitys-metodin ja poisto-metodin.

Rakentamisen yhteydessä SceneManager-olio tekee Three.js-kirjaston näkymän ja upottaa siihen CanvasEntryPoint-oliolta saaman canvas-elementin. Tämän jälkeen ohjelma odottaa käyttäjää valitsemaan halutun ajoneuvon tiedoston. Käyttäjän valittua tiedoston SceneManager-olio alkaa tehdä näkymältä vaadittuja SceneSubject-olioita kuten esimerkiksi ajoneuvoa, ajoneuvon varsistoa, ulottumakäyrää oletusasennossa, maata, mitta-asteikkoa ja valaistusta.

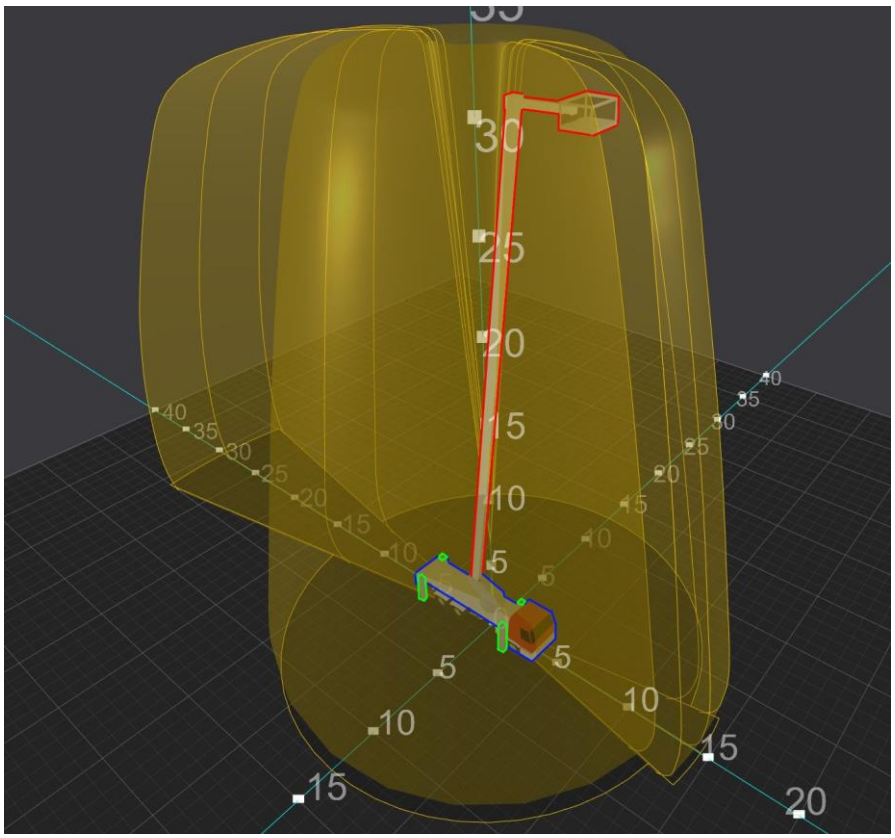
Oletusasetusten rakentamisen jälkeen käyttäjä on vapaa säätämään käyttöliittymän eri asetuksia, ja SceneManager-olio muokkaa näkymän osia näiden asetusten perusteella. Jokainen käyttäjän aiheuttama muutos Redux-kirjaston tietokantaan laukaisee ”handleChange” nimisen SceneManager-luokan

metodin, joka poistaa kaikki näkymän SceneSubject-oliot ja rakentaa ne uudelleen uusien asetusten mukaan.

SceneManager-luokan alkuperäinen ajatus oli olla yksinkertainen, mutta siitä paisui kaiken kattava monoliittinen osa ohjelmaa. Ongelmana on, että luokka osittain itse laskee osien uudet paikat ja muodot sen sijaan, että jakaisi syötteet osille ja antaisi niiden itse laskea laskettavansa. Ennen kuin ongelmasta tuli huomattava, SceneManager-luokka oli jo paisunut ja monimutkaistunut niin, että sen uudelleen jäsennöinti oli hankalaa.

4.3.3 ModelLoader + Crane + HLACrane + SceneOutrigger

Näkymällä esiintyvä ajoneuvon malli koostuu kolmesta eri osasta: ModelLoader-olio eli ajoneuvon malli, Crane-olio tai HLACrane-olio eli ajoneuvon varsisto ja SceneOutrigger-olio eli ajoneuvon tukijalat. Kuvassa 3 näkyy ModelLoader-olio sinisellä rajattuna, Crane-olio punaisella rajattuna, sekä SceneOutrigger-olio vihreällä rajattuna.



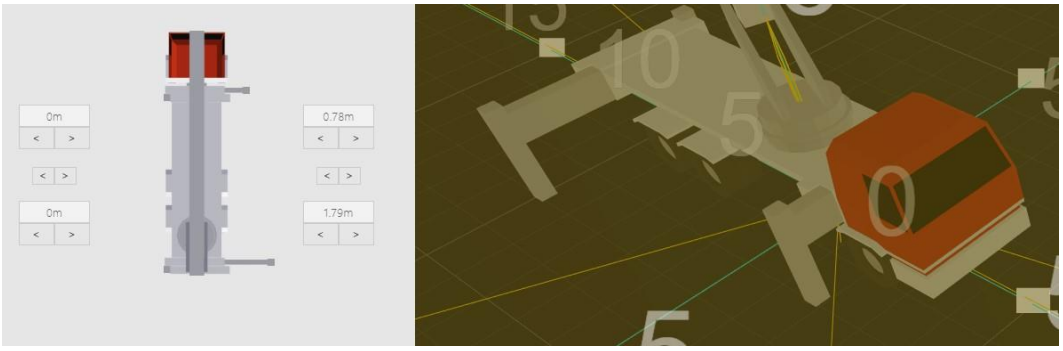
KUVA 3. Ajoneuvon kolme osaa eri värein rajattuna

Ajoneuvoja voi ohjelmassa olla kahdenlaisia, normaaleja tai HLA-laitteita. HLA-laitteet eroavat normaaleista siinä, että ne ovat huomattavasti normaalia laitetta korkeampia ja suurempia. HLA-laitteiden varsisto on myös erilainen verrattuna normaaliin laitteeseen.

ModelLoader-olio on yksinkertaistettu kolmiulotteinen malli eräästä Bronto Skyliftin oikeasta ajoneuvosta. ModelLoader-oliolla on käytössään kaksi mallia riippuen siitä, onko laite normaali vai HLA-laite. Malli on täysin staattinen ohjelman aikana eikä muutu tiedoston lataamisen jälkeen.

Crane- tai HLACrane-olio on ajoneuvon varsisto. Varsiston pituus ja kulma perustuvat ajoneuvotiedostossa esiintyviin arvoihin, eivätkä ne muutu tiedoston lataamisen jälkeen. Siinä missä ModelLoader-olio lataa valmiiksi tehdyn kolmiulotteisen mallin, Crane-olio joutuu tekemään varsiston osan dynaamisesti. Normaali varsisto koostuu neljästä eri osasta, pitkästä teleskooppivarresta, korivarresta, työkorista, sekä nivelistä kaikkien osien välillä. Varsiston eri osat rakennetaan Three.js-kirjaston Shape-luokan olioilla, asetetaan oikeisiin paikkoihin, liitetään toisiinsa ja lopuksi asetetaan osien kulmat. Viimeiseksi ladataan varsiston päässä oleva työkorin kolmiulotteinen malli, joka liitetään korivarteen. Lopuksi varsisto liitetään jalustan ja kotelorungon kautta kiinni ajoneuvoon. Varsisto on ainoa SceneSubject-olio ohjelmassa, jolla on aiemmin mainittu SceneSubject-luokasta peritty päivitys-metodi. Metodilla varsisto pyörii alustansa ympäri osoittaen aina käyttäjän osoittamaan kohtaan ulottumakäyrällä.

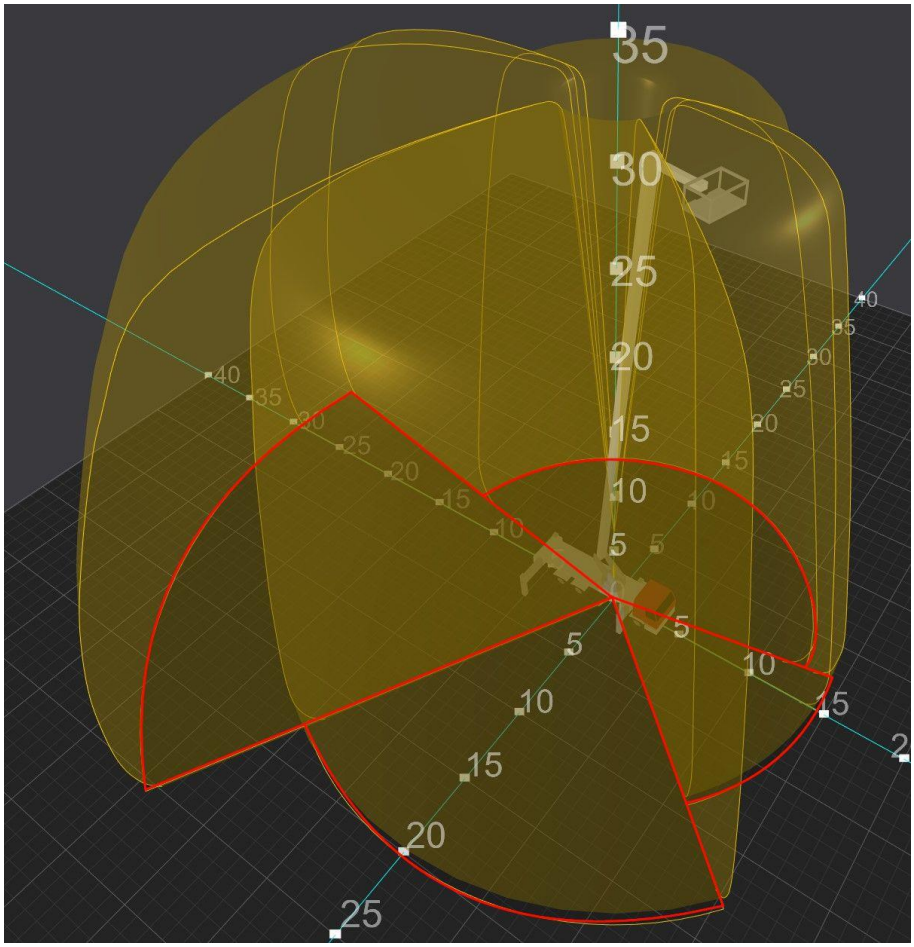
SceneOutrigger-luokan oliot ovat ajoneuvossa esiintyvät tukijalat. Tukijalat tehdään kuten Crane-luokan varsisto-oliotkin. Tukijalkojen asento riippuu tukijalkojen asetuksista käyttöliittymässä kuten kuvassa 4 näkyy.



KUVA 4. Tukijalkojen asento riippuu tukijalkojen asetuksista

4.3.4 Arc

Arc-olio on SceneSubject-luokan olio, jolla voi piirtää ympyrän kaaren tai sektorin. Ohjelmassa sitä käytetään näyttämään ulottumakäyrän suurin mahdollinen ulottuma. Kaaren ulottuma riippuu käyttäjän valitsemien tukijalkojen asennoista. Kuvassa 5 on erikokoiset Arc-oliot korostettu punaisella.



KUVA 5. Erikokoisia sektoreita

Arc-oliot ja muutkin ulottumakäyrän oliot koostuvat kahdesta osasta, vahvistetusta muodon ääriviivasta ja itse muodosta (eng. shape). Käyttäjän syötteen tullessa SceneManager-olio hakee jokaisen ulottumakäyrän sektorin halkaisijat, kulman alun ja lopun ja laskee syötetyt arvot näkymälle sopiviksi arvoiksi ja rakentaa Arc-olion.

Arc-olio tiedot saatuaan rakentaa Three.js-kirjaston Shape-luokan olion ja siirtää sen origosta y-akselia myöten eteenpäin. Tässä pisteessä muodosta tehdään kaari käyttäen Three.js-kirjaston ”absarc”-komentoa, joka origon, halkaisijan ja kulmien tiedoilla tekee oikean kokoisen kaaren. Kaareen lisätään suora viiva käyttäen Three.js-kirjaston ”lineTo”-komentoa origon suuntaan, joka luo suoran viivan tehdyn kaaren lopusta origoon. Lopuksi kutsutaan ”closePath”-komentoa, joka sulkee kaaren tekemällä suoran viivan kaikkien pisteiden välillä, jotka ovat vapaita, eli tässä tapauksessa origon ja kaaren alkupisteen välillä luoden kaaren.

Shape-olioista tehdään Three.js-kirjaston ShapeGeometry-luokan olio, sillä pelkkä Shape-olio ei riitä näkymällä piirrättämiseen. ShapeGeometry-olion rinnalla rakennetaan Three.js-kirjaston MeshBasicMaterial-luokan olio, joka kertoo Three.js-kirjastolle miltä Shape-luokan olion kuuluu näyttää piirrettäessä. Lopuksi ShapeGeometry-olio ja MeshBasicMaterial-olio yhdistetään Three.js-kirjaston Mesh-luokan olioksi, joka lisätään näkymälle piirrettäväksi. Lopuksi tehdään ääriviivan antava olio käyttäen aiemmin rakennettua ShapeGeometry-oliota Three.js-kirjaston Line-luokan oliossa, joka myös lisätään näkymään.

4.3.5 CatmullRomCurve

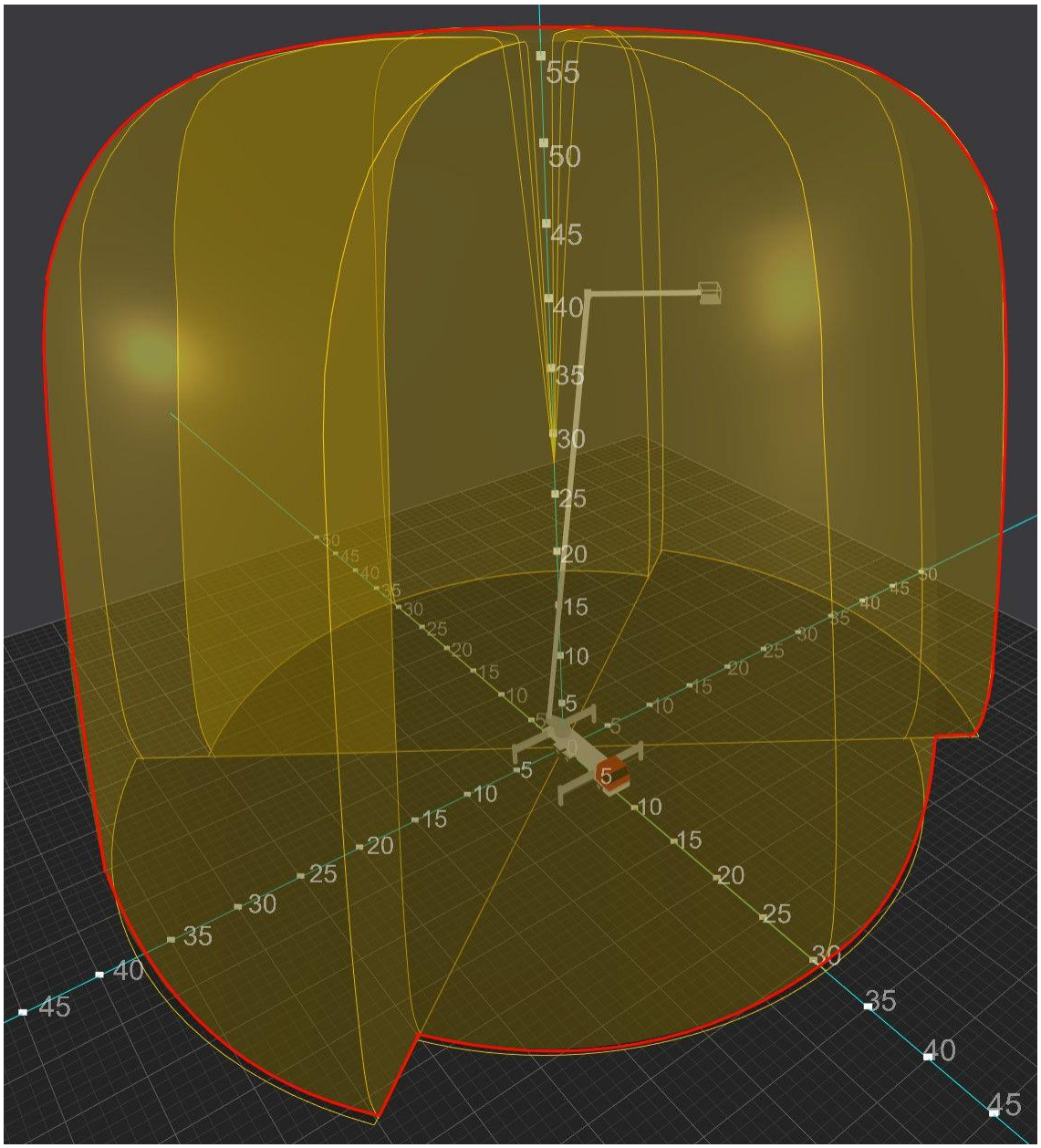
CatmullRomCurve-luokka alun perin periytyi SceneSubject-luokasta, mutta projektin edetessä osasta tuli yksinkertaisempi itsenäinen osa, jota ei tarvitse piirtää näkymällä.

CatmullRomCurve-luokka on yksinkertainen kääreluokka (eng. wrapper) Three.js-kirjaston CatmullRomCurve3-luokalle. Three.js-kirjaston luokka tekee pehmeän kolmiulotteisen käyrän käyttäen Catmull-Rom-algoritmia ja pistejonoa. Catmull-Rom-algoritmi on algoritmi, joka tuottaa splini-käyrän pistejonosta.

Ajoneuvojen tiedostoissa on määritelty jokaisen eri ajoneuvon asetuksen ja sektorin kohdalle pistejono, joka vastaa ulottumakäyrän muotoa niillä asetuksilla. SceneManager-olio hakee nämä tiedot ja tekee jokaiselle sektorille oman käyrän käyttäen CatmullRomCurve-luokan olioita ja tallentaa käyrän tiedot CatmullRomCurveDome-luokan olioita varten.

4.3.6 CatmullRomCurveDome + CustomCurve

CatmullRomCurveDome-luokan olio on käyttäjän kannalta ohjelman olennaisin osa, eli ajoneuvon varsiston ulottumakäyrä. Kuvassa 6 esiintyy ulottumakäyrä punaisella korostettuna.

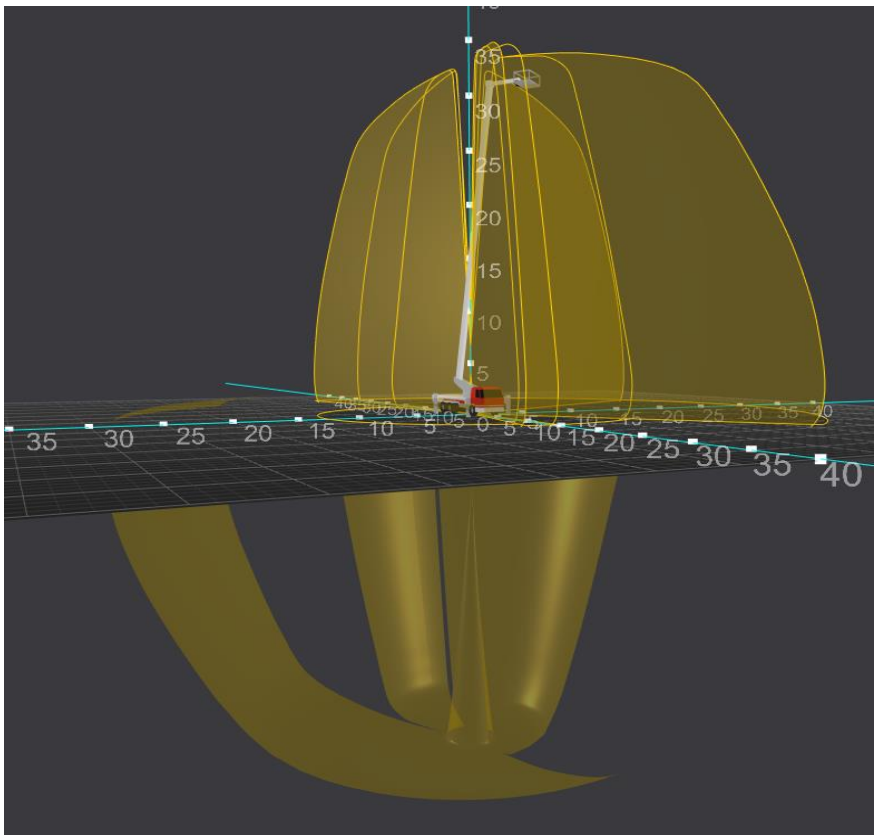


KUVA 6. Ulottumakäyrä

Ulottumakäyrä on CatmullRomCurve-luokan olion käyrä, joka pursotetaan (eng. extrude) Arc-olion kaaren mukaan. Lopputuloksena on kolmiulotteinen sektori, jonka sivuleikkaus on CatmullRomCurve-olion käyrän muotoinen.

Ulottumakäyrää tehtäessä käytetään CustomExtrudeGeometry nimistä luokkaa. Tämä luokka pursottaa halutun muodon halutuilla asetuksilla, esimerkiksi jonkin käyrän mukaan.

Aluksi tehdään käyrä käyttäen CustomCurve-luokkaa. CustomCurve-luokka on kopio Three.js-kirjaston Curve-luokasta, mihin on tehty pieni käyrän normaaleja koskeva säätö. Käyrän normaalilla tässä tapauksessa tarkoitetaan vektoria joka leikkaa tasossa kohtisuorasti käyrän. Three.js-kirjasto on tehty tukemaan nimenomaan kolmiulotteisia toteutuksia, joten Curve-luokan käyrien normaalit voivat myös osoittaa kolmiulotteisesti. Tässä tapauksessa kuitenkin ulottumakäyrän muoto pitäisi pursottaa tasoa vasten olevan kaksiulotteisen käyrän mukaan ja vektorin suunnan pitäisi olla ulottumakäyrästä ulospäin. Jos Three.js-kirjaston Curve-luokkaa käyttää niin ulottumakäyrä suorastaan räjähtää kuten kuvassa 7 on esitetty.



KUVA 7. Räjähtänyt ulottumakäyrä

Räjähtänyt ulottumakäyrä johtuu Curve-luokan kaaresta, joka laskee käyrän normaalit sisäänpäin ja ylösalaisin. Normaalin laskemista ei voi pelkällä Curve-luokan rakentajalla vaihtaa, joten koko luokka piti kopioida ja vaihtaa pieni osa luokan koodista, jossa normaalit lasketaan.

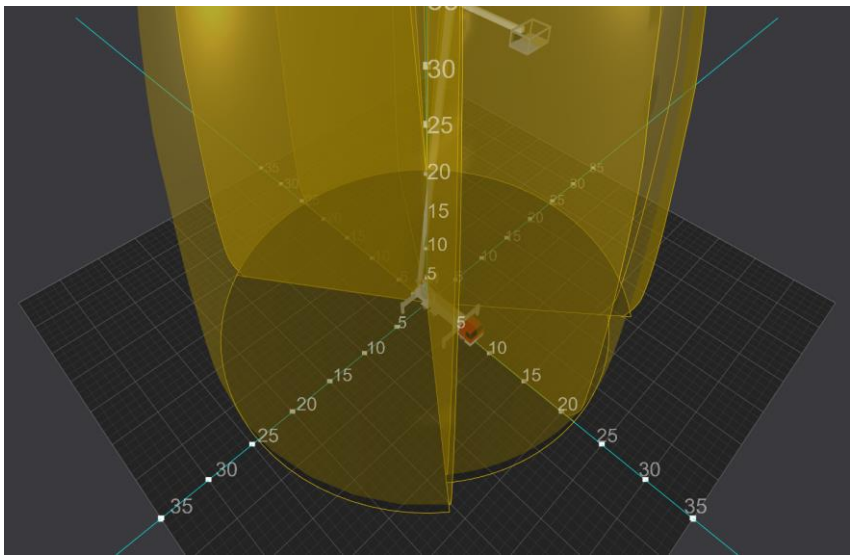
Kun CustomCurve-luokan olio on rakennettu, annetaan sille ”getPoint” niminen metodi, joka syötteeksi ottaa arvon nollasta yhteen ja palauttaa vastaavan pisteen halutun sektorin Arc-olion kaaresta. Metodia käyttää CustomExtrudeGeometry-luokan olio pursottaessaan ulottumakäyrää.

Lopuksi tehdään CustomExtrudeGeometry-luokan olio, jolle annetaan arvoksi CatmullRomCurve-olion käyrän muoto, CustomCurve-olion käyrä ja ”steps” niminen arvo, jolla määritetään, kuinka pehmeä muodon tulee olla. Jos steps-arvo on yksi, on muoto kolmio, kun taas steps-arvolla 100 muoto on kutakuinkin ympyrän kaaren muotoinen. Viimeiseksi oliolle annetaan kiiltävä materiaali ja se viedään näkymälle piirrettäväksi.

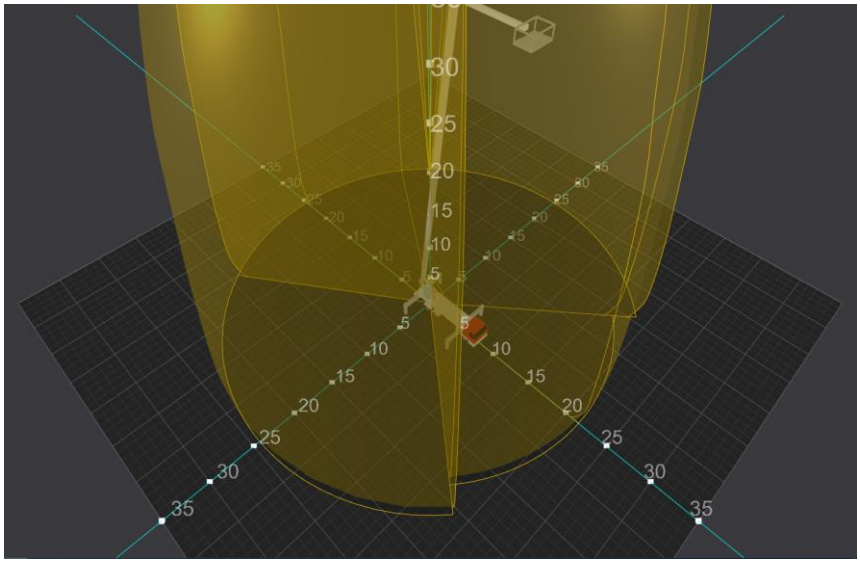
4.3.7 CustomExtrudeGeometry

Kuten CustomCurve-luokka, myös CustomExtrudeGeometry-luokka on kopio Three.js-kirjaston luokasta pienin säädöin. Toisin kuin CustomCurve-luokan kanssa nämä säädöt ovat kosmeettisia, mutta käyttäjämukavuuden kannalta tärkeitä.

Kun Three.js-kirjaston ExtrudeGeometry-luokka pursottaa muotoa käyrää pitkin, se lopussa syntyneelle muodolle rakentaa pohjan ja päällisen. Tässä toiminnallisuudessa ei ole mitään vikaa, mutta jos muotoja on monta samaan aikaan lähekkäin ja niiden materiaali on läpikuultavaa, on lopputulos läpikuultamaton. Kuvassa 8 näkyy ulottumakäyrä ExtrudeGeometry-luokan oliolla ja kuvassa 9 on samojen asetusten ulottumakäyrä CustomExtrudeGeometry-luokan oliolla.



KUVA 8. Three.js-kirjaston ExtrudeGeometry-luokan olio



KUVA 9. CustomExtrudeGeometry-luokan olio

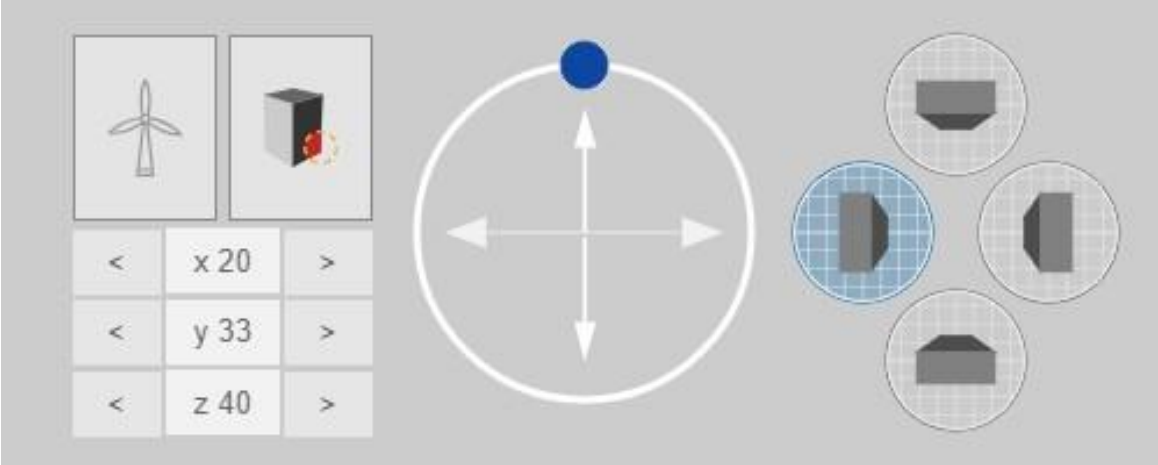
Three.js-kirjaston ExtrudeGeometry-luokan tulos on huomattavasti läpikuultamattomampi ja käyttäjän on vaikeampi erottaa esimerkiksi mitta-asteikon lukemia ulottumakäyrän takaa. Koska ExtrudeGeometry-luokka tekee jokaiselle muodolle aina päällisen ja pohjan kahden eri sektorin leikkauspisteessä on sekä päällinen että pohja käytännössä samassa pisteessä, josta aiheutuu näkymän läpikuultamattomuus. CustomExtrudeGeometry-luokka ratkaisee asian lisäämällä luokan rakentajaan arvot, joiden perusteella pohja tai päällinen joko tehdään tai ei tehdä. SceneManager-olio ennen CatmullRomCurveDome-olion tekemistä päättelee pitäisikö kyseisellä sektorin ulottumakäyrällä olla pohja, päällinen vai molemmat tai ei ollenkaan, ja sitten syöttää nämä tiedot

CatmullRomCurveDome-oliolle, joka vie ne eteenpäin CustomExtrudeGeometry-luokalle.

4.3.8 Building + Windmill

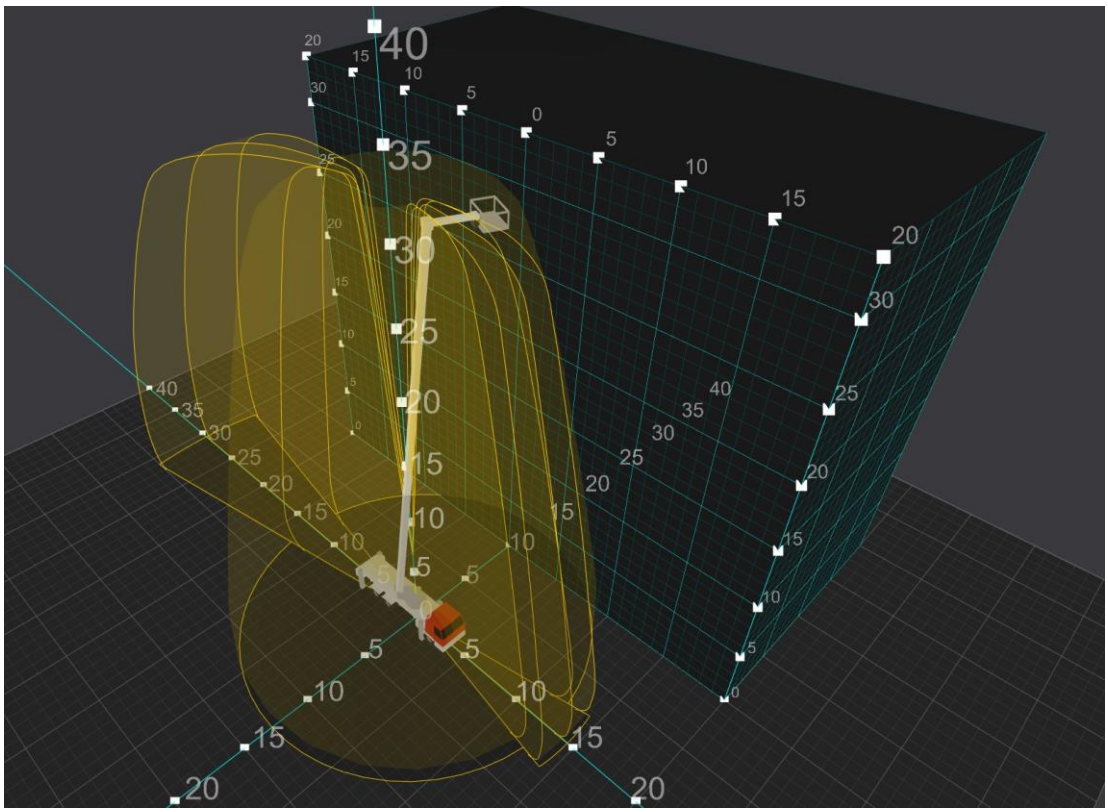
Building on SceneSubject-luokan olio, joka esittää rakennusta. Nostolavojen tarkoitus on nostaa ihmisiä ja rakennukset ovat tyypillisiä paikkoja, joihin heitä nostetaan, joten käyttäjälle on tärkeää tietää kuinka pitkälle valitun ajoneuvon ulottumakäyrä ulottuu rakennukseen verrattuna.

Kuvan 10 käyttöliittymän avulla käyttäjä pystyy valitsemaan rakennukselle mielivaltaisen koon, paikan ja kulman. Käyttäjä pystyy myös valitsemaan neljästä oletusasennosta, sekä vaihtamaan rakennuksen tuulimyllyksi. Tuulimyllyssä ei voi säätää tuulimyllyn leveyttä, sen sijaan lapojen kulmaa voi vaihtaa. Käyttöliittymästä voi myös laittaa ulottumakäyrän ja rakennuksen välisten leikkauspisteiden näkymän päälle.

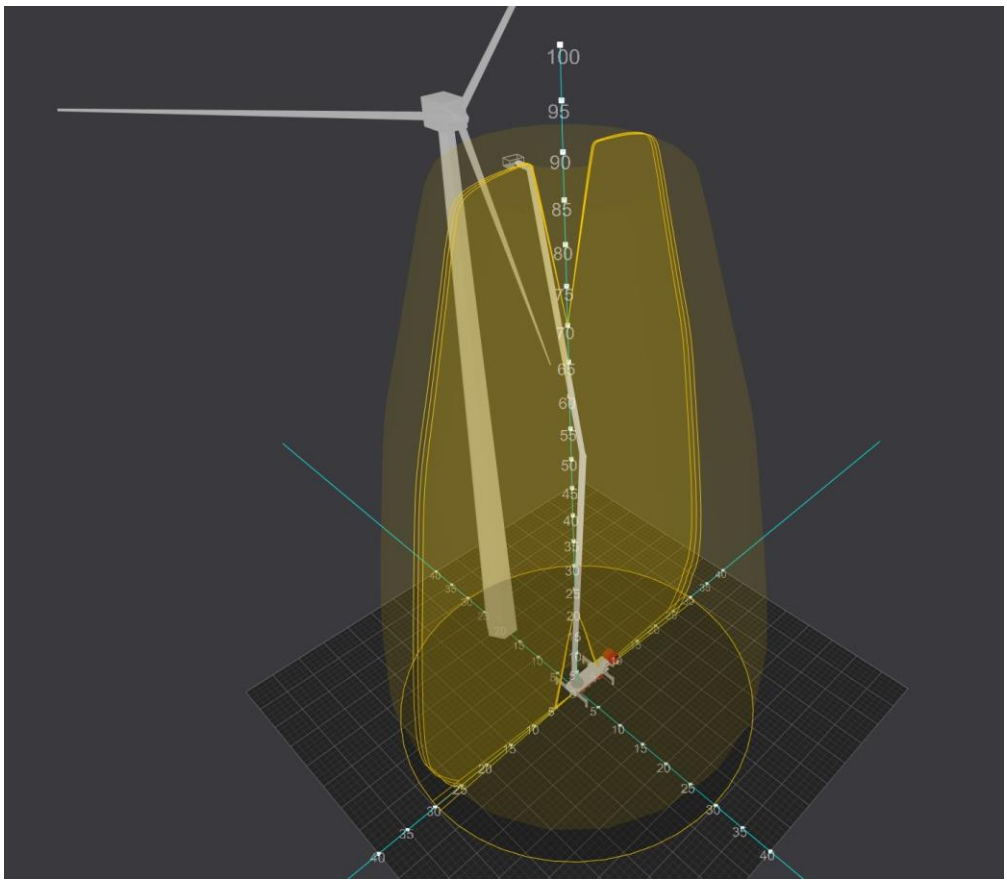


KUVA 10. Rakennuksen käyttöliittymä

Kuvassa 11 näkyy oletusasetusten rakennus ja kuvassa 12 näkyy oletusasetusten tuulimylly, sekä HLA-laite.



KUVA 11. Oletusasetusten rakennus



KUVA 12. Oletusasetusten tuulimylly, sekä HLA-laite

Normaali rakennus on yksinkertainen Three.js-kirjaston Shape-luokan olio, Crane-luokan olioiden tapaan, mutta eri kokoinen. Tuulimylly on myös Crane-luokan olioiden tapaan luotu, mutta paljon monimutkaisempi. Koska tuulimyllyn korkeutta ja lavojen leveyttä ja kulmaa voi säätää, pitää kaiken tuulimyllyssä olla dynaamisesti rakennettu.

Rakennusprosessi tuulimyllylle on samanlainen kuin Crane-luokassa, ensin tehdään kaikki muodot, jotka muutetaan geometrioiksi ja sitten liitetään yhteen. Ero Crane-luokkaan tulee liittämisen jälkeen, jolloin Three-csg-luokan avulla liitetyt geometriat tehdään yhdeksi ja samaksi geometriaksi. Koska tuulimylly on rakennus ja sen pitää pystyä näyttämään leikkauspisteet ulottumakaavion kanssa, pitää siitä tehdä yksi geometria monen sijaan. Jos tuulimylly jätettäisiin moneen osaan, pitäisi leikkauspisteitä laskettaessa laskea jokainen sektori jokaista tuulimyllyn osaa vasten, joka vaatisi enemmän suoritustehoa ja aikaa, kuin jokainen sektori yhtä tuulimyllyä vasten.

Tuulimyllyn lavat eivät ole täysin suorakaiteen muotoisia, vaan ne kapenevat leveys- ja syvyyssuunnassa kärkeä kohti, kuten kuvassa 12 esitetään.

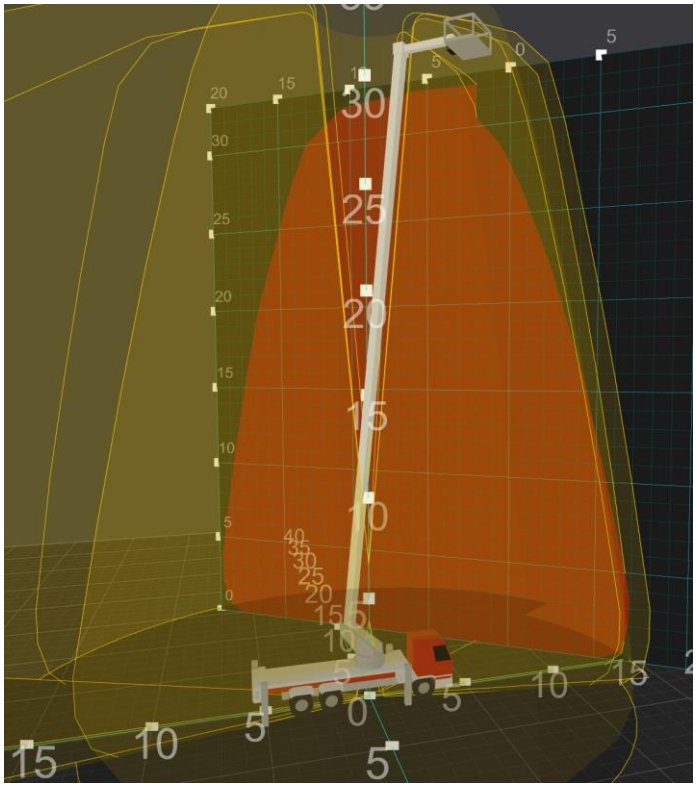
KUVA 12. Tuulimyllyn kapenevat lavat

Three.js-kirjastossa ei ole valmista helppoa tapaa muokata luotujen kolmiulotteisten mallien olioiden muotoja halutulla tavalla. Lavat on toteutettu tekemällä ensin tavanomainen Three.js-kirjaston BoxGeometry-luokan olio ja antamalla sille halutut arvot, tämän jälkeen geometriasta tarvittavat kärjet (eng. vertex) siirretään hieman sisäänpäin. Lopuksi geometriassa oleva verticesNeedUpdate-arvo asetetaan todeksi, jotta Three.js-kirjasto osaa päivittää geometrian vastaamaan haluttua geometriaa.

4.3.9 Three-csg

Chandler Prallin (2019) tekemä Three-csg-kirjasto on kolmannen osapuolen kirjasto Three.js-kirjastoa varten, joka mahdollistaa geometrioiden lisäämisen ja poistamisen toisistaan, sekä geometrioiden leikkauspisteiden havainnoinnin. CSG on lyhenne englanninkielen sanoista Constructive Solid Geometry.

Käyttäjän laittaessa leikkauspisteet päälle, ulottumakäyrän ja rakennuksen välinen leikkaus näkyy erittäin voimakkaasti, kuten kuvassa 13 näkyy.

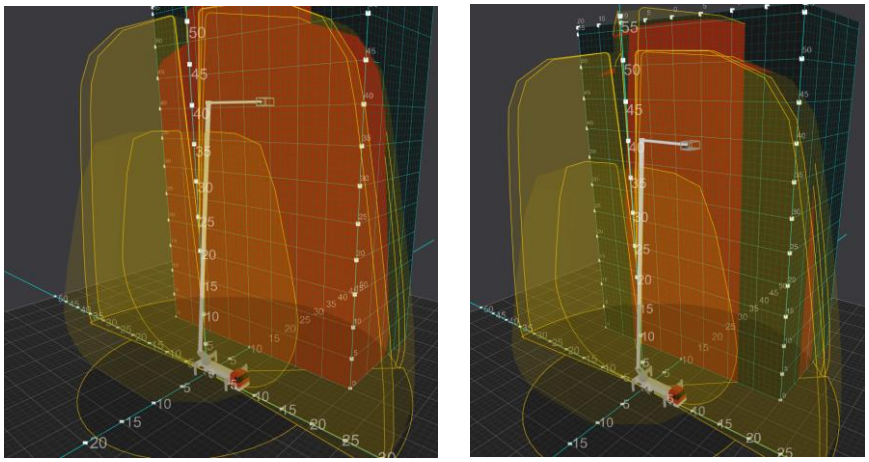


KUVA 13. Rakennuksen ja ulottumakäyrän leikkaus

Leikkauspisteiden ollessa päällä SceneManager-olio hakee rakennetut ulottumakäyrät ja vertaa kaikkien käyrien geometrioita rakennuksen geometriaan käyttäen Three-csg-kirjaston ”intersect” nimistä metodia. Metodi palauttaa geometrioiden välisen leikkauksen uutena geometriana, joka lisätään näkymään. Lopuksi talosta pitää poistaa leikkauksen muotoinen geometria, sillä muuten leikkauksen geometria ja talon geometria eivät piirtyisi oikein.

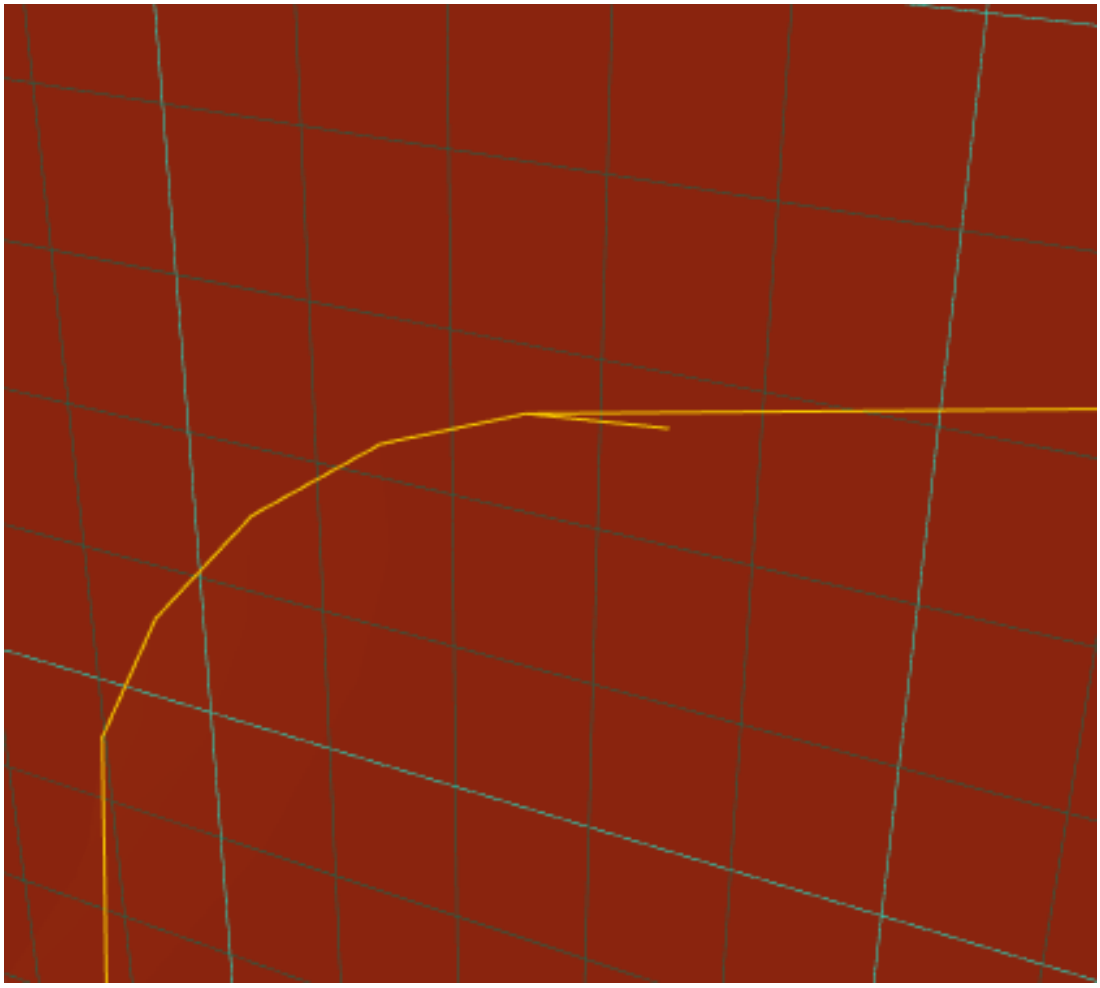
Leikkauksen laskeminen on matemaattisesti raskas toiminto tietokoneelle, joten leikkauspisteiden ollessa päällä SceneManager-olio laskee CatmullRomCurveDome-luokan steps-arvoa. Steps-arvon laskeminen tekee ulottumakäyrän geometriasta yksinkertaisemman vähentäen matemaattisen laskun laskemisen määrää. Ulottumakäyrän tarkkuus kärsii steps-arvon laskiessa, mutta ei kuitenkaan tee leikkauksen havainnoinnista liian epätarkkaa.

Three-csg-kirjastoa ja muita vastaavia metodeja käyttäessä täytyy olla tarkkana verrattavien olioiden geometrioista. Jos geometriassa on pienikin virhe, koko lasku antaa väärän tuloksen ja leikkauksesta tulee vajaa tai jopa täysin satunnainen. Kuvassa 14 on kaksi eri tilannetta, joissa molemmissa on sama laite ja rakennus samoilla asetuksilla. Vasemmanpuoleisen kuvan tilanteen jälkeen on ulottumakäyrän käyrän laskua muokattu niin, että käyrän ylemmän osan laskeminen on aloitettu kaksi pistettä liian aikaisin, jolloin geometriaan tulee ihmisen silmään erittäin vaikeasti havaittava pieni poikkeama. Poikkeama aiheuttaa kuitenkin huomattavan ja epämääräisen muutoksen ulottumakäyrän ja rakennuksen leikkauksessa.



KUVA 14. Saman tilanteen leikkaus hieman poikkeavilla geometrioilla

Kuvassa 15 näkyy käyrän pieni poikkeama, joka aiheutti virheellisen tilanteen.

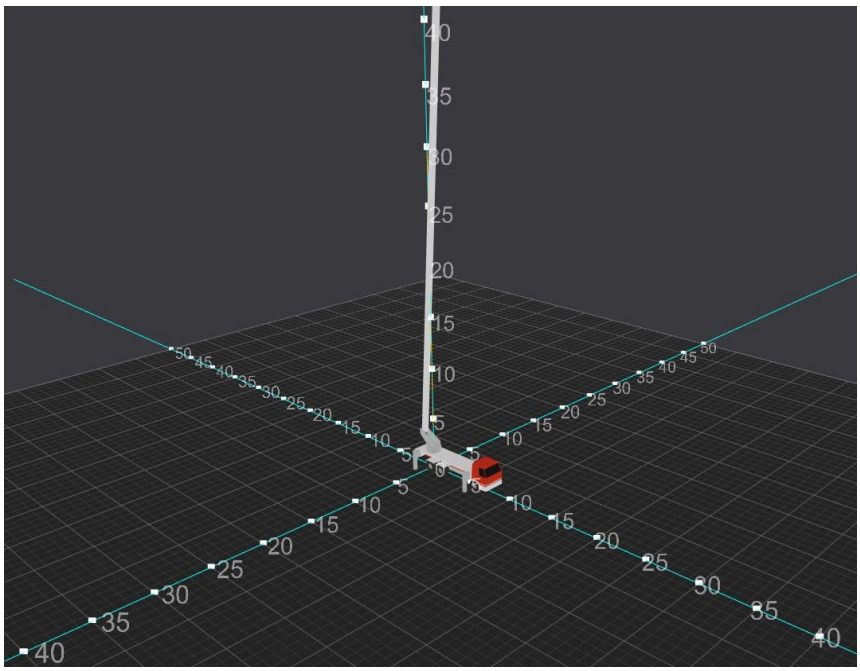


KUVA 15. Poikkeama käyrässä

Poikkeaman voi havaita vain suurentamalla kameraa tarpeeksi käyrää kohti tai käyttämällä ohjelmasta löytyvää kaksiulotteista kameraesitystä.

4.3.10 GridHelper + CustomGridHelper

GridHelper-olio on SceneSubject-luokan olio, joka käyttää Three.js-kirjaston samannimistä luokkaa, sekä Three.js-kirjaston Sprite-luokkaa tehdäkseen ruudukon ja mitta-asteikon. Kuvassa 16 näkyy mitta-asteikko ja ruudukko.

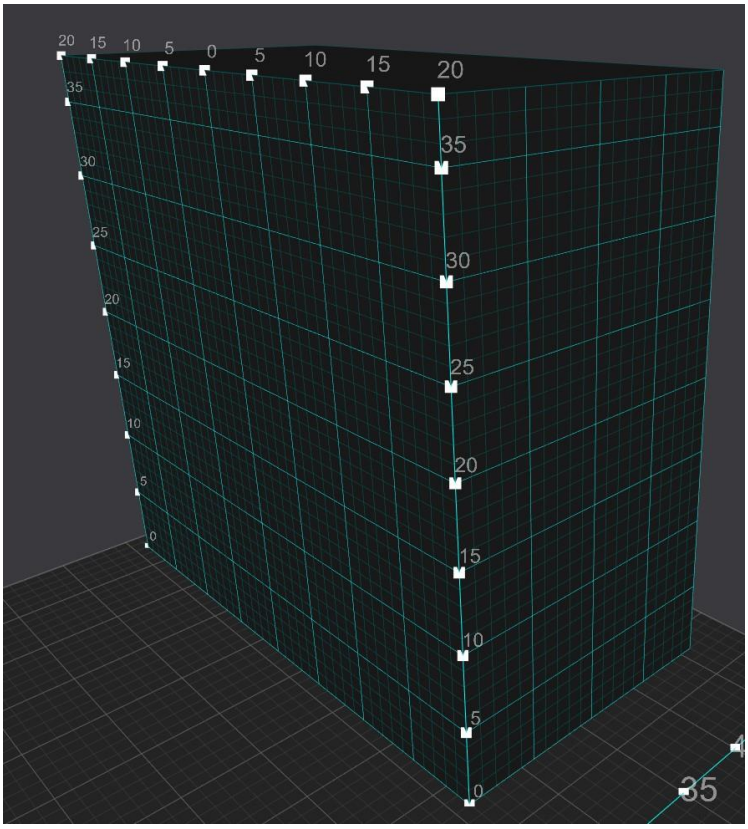


KUVA 16. Ruudukko ja mitta-asteikko

Ruudukko on tehty käyttäen kahta Three.js-kirjaston GridHelper-luokan oliota. Toisen olion ruutujen väliksi on määritelty viisi, joka ohjelmassa vastaa viittä metriä, ja toisen olion ruutujen väli on yksi metri. Ruudukoille annetaan kooksi kaikkien valitun ajoneuvon ulottumakäyrien mahdollisten leveyksien suurin arvo, johon lisätään kymmenen metriä jokaiselle sivulle. Ruudukot lopuksi asetetaan limittäin ja näkymän keskelle.

Mitta-asteikon tekeminen ja kirjaimien esittäminen näkymässä ylipäätään on hankalaa Three.js-kirjastoa käyttäen. Three.js ei valmiiksi tue kirjaimien kirjoittamista suoraan, joten seuraavaksi paras ratkaisu on käyttää Three.js-kirjaston Sprite-luokkaa. Sprite-luokka tarvitsee toimiakseen Three.js-kirjaston Texture-luokan tekstuurin ja SpriteMaterial-luokan materiaalin. Texture-luokan rakentaja ottaa yhdeksi mahdolliseksi syötteeksi HTML5-canvas-elementin, joten mitta-asteikkoa tehtäessä jokainen numero luo HTML5-canvas-elementin, johon numero piirretään ja canvas sitten syötetään Texture-luokan rakentajaan. Lopuksi Texture-luokan olio annetaan syötteeksi Sprite-luokalle, läpikuultavan SpriteMaterial-luokan olion kanssa. Lopullinen Sprite-olio asetetaan oikeaan kohtaan näkymällä pienen Three.js-kirjaston Points-luokan olion kera. Sprite-luokan oliot eivät ole kolmiulotteisia ja näkyvät aina kameraan päin, joten käyttäjän ei tarvitse säätää kameraa nähdäkseen mitta-asteikkoa.

Ruudukko ja mitta-asteikko ovat kiinni myös rakennuksessa, kuten kuvassa 17 näkyy. Mitta-asteikko tehdään kuten maata vasten olevassa mitta-asteikossakin, mutta ruudukon kanssa on käytetty CustomGridHelper-luokkaa. Three.js-kirjaston normaali GridHelper-luokka ottaa syötteekseen vain koon ja ruudukkovälin, mutta rakennus voi olla muutakin kuin neliön muotoinen. CustomGridHelper-luokka lisää ominaisuuden valita ruudukon koon normaaliin GridHelper-luokkaan.



KUVA 17. Ruudukko ja mitta-asteikko rakennuksessa

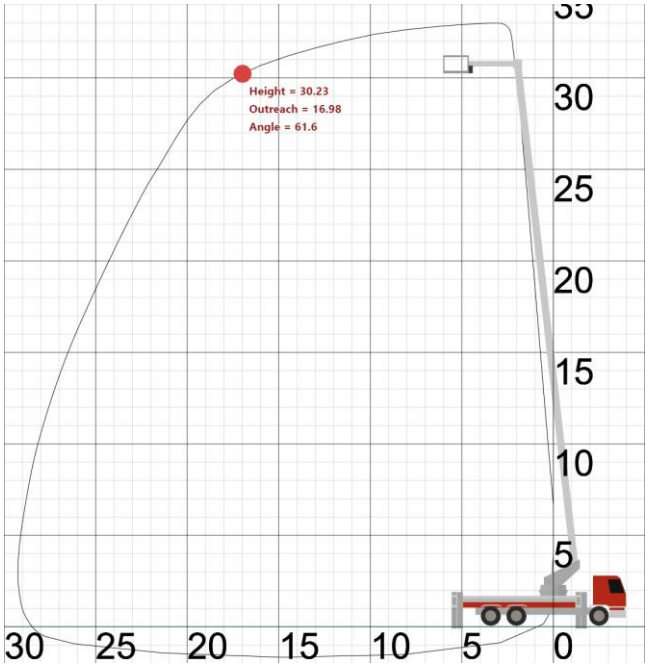
4.3.11 Ortografinen kameraesitys

Three.js kirjastolla on kaksi erilaista kameraesitys mahdollisuutta, joko normaali perspektiivikamera (PerspectiveCamera) tai ortografinen-kamera (OrthographicCamera). Tähän mennessä jokainen kuva opinnäytetyössä on otettu ohjelman perspektiivikameralla.

Three.js-kirjaston dokumentaation mukaan perspektiivinen projektio ”imitoi ihmisen silmän tapaa nähdä” (Three.js 2019). Perspektiivisen projektion tarkoitus

on pienentää kauempana kamerasta olevia asioita ja suurentaa lähempänä kameraa olevia asioita, lopputuloksena on syvyyden vaikutelma kaksiulotteiselta tietokoneen ruudulta. Ortografisessa projektiossa kaikki asiat pidetään sen kokoisena kuin ne ovatkin huolimatta niiden etäisyydestä kameraan, mikä antaa kuvalle kaksiulotteisen vaikutelman, vaikka näkymällä on vain kolmiulotteisia esineitä.

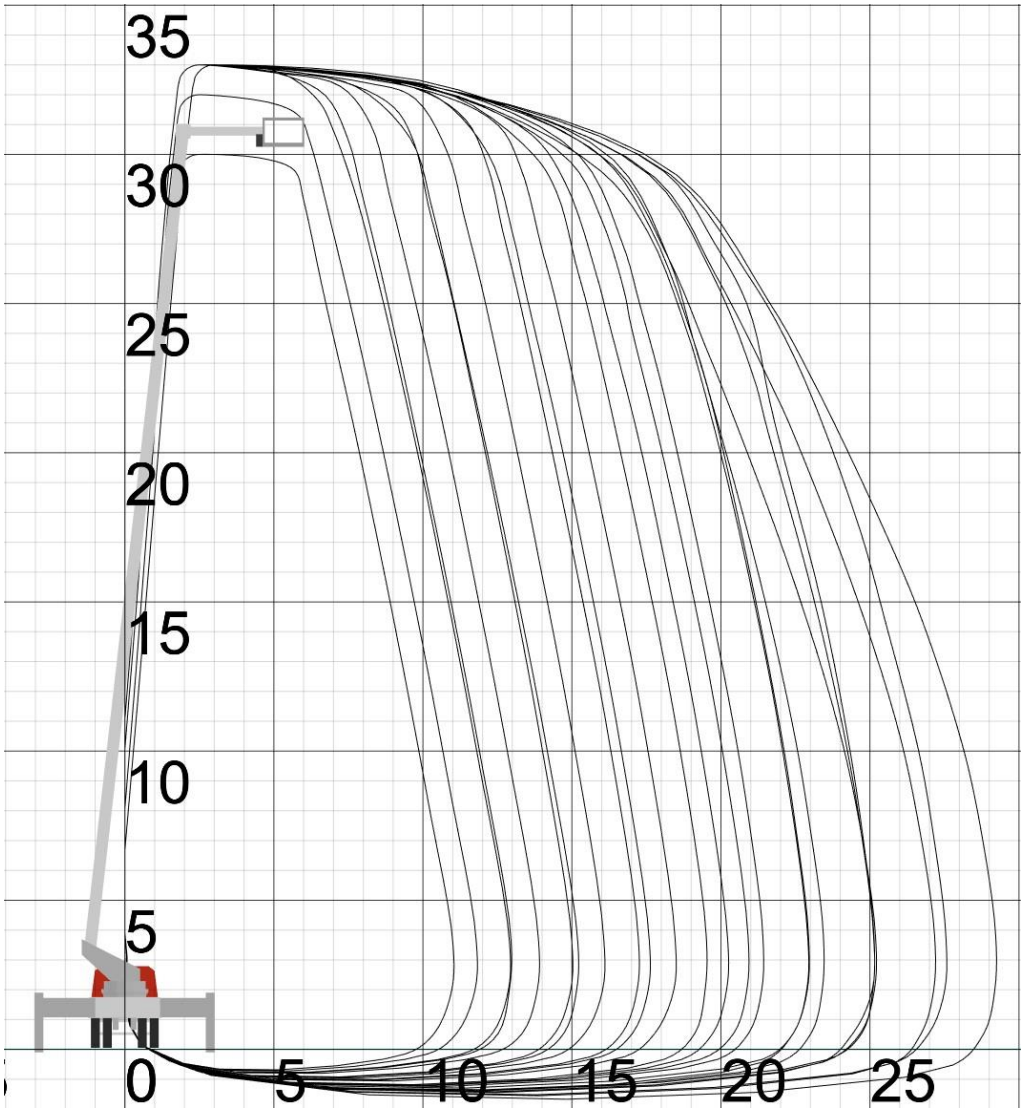
Ohjelmassa ortografista kameraa käytetään vaadittuun kaksiulotteiseen esitystapaan, jossa ulottumakäyriä voisi tutkia eri asennoista tarkemmin kuin mitä perspektiivi kameralla pystyy. Ortografisen esityksen saa päälle käyttöliittymän nappia painamalla. Kuvassa 18 on ortografinen esitys ulottumakäyrästä ajoneuvon takaa sivultapäin kuvattuna.



KUVA 18. Ortografinen esitys

Ortografinen-esitys käyttää samaa näkymää kuin normaali perspektiiviesityskin, mutta SceneManager-olio ei anna CatmullRomCurveDome-luokan olioille mitään kaarta mitä pitkin pursottaa, joten ruudulle piirtyy vain ulottumakäyrän ääriviiva. Ortografisessa esityksessä käyttäjä ei voi vapaasti vaihtaa kameran kuvakulmaa, vaan käyttöliittymässä on annettu vaihtoehtona muutama kuvakulma, joita painamalla kuvakulma vaihtuu. Kuvakulman vaihtuessa SceneManager-olio piirtää vain valitulle kuvakulmalle olennaisia ulottumakäyriä, jotta käyttäjä voi tutkia vain valittua ulottumakäyrää.

Ortografisessa esityksessä on myös muutama erikoistoiminto, joita perspektiiviesityksessä ei ole. Kuvassa 19 näkyy yksi erikoistoiminto, jolla ajoneuvon kaikkien mahdollisten asetusten ulottumakäyrät näkyvät samanaikaisesti.



KUVA 19. Kaikki ulottumakäyrät samanaikaisesti

5 POHDINTA

Opinnäytetyön tarkoituksena oli tehdä Bronto Skyliftin vaatimusten mukainen kolmiulotteinen näkymä nostolavojen ulottumien etäisyyksien, korkeuksien ja muotojen havainnointiin. Tuloksena on moderni selainpohjaisiin teknologioihin perustuva kolmiulotteinen näkymä, jonka rakennetta ja osia opinnäytetyö kuvailee.

Suurin osa kolmiulotteisen näkymän rakenteesta, osista ja logiikasta on omaa käsialaani, mikä johti opinnäytetyössä mainittuun SceneManager-luokan paisumiseen ja muihin paikallisiin hankaluuksiin. Työn kolmiulotteista näkymää tehdessä Three.js-kirjaston dokumentaatio oli useaan otteeseen riittämätön, mikä johti kirjaston lähdekoodin tutkimiseen ja analysointiin. Äärimmäisissä tapauksissa Three.js-kirjaston luokka, kuten CustomCurve-luokka, piti kopioida ja muokata, jotta vaatimukset täyttyisivät. Hankaluuksista huolimatta, lopputuotos on mielestäni, sekä asiakkaan palautteen mukaan, erittäin onnistunut, etenkin kun vastaavaan kolmiulotteiseen toteutukseen ei ollut valmiita ohjeita tai esimerkkejä.

Jos voisin tehdä jotain toisin, niin suunnittelisin näkymän rakenteen paremmin. Mahdollisen jatkokehityksen kannalta SceneManager-luokka tulee hidastamaan kehitystä ja aiheuttamaan hankaluuksia, etenkin jos en ole itse jatkokehittämässä.

LÄHTEET

Bronto Skylift. 2019. Verkkoaineisto. Luettu 13.4.2019. https://brontoskylift.com/

Chandler, P. 2019. Constructive Solid Geometry library for Three.js. Verkkoaineisto. Luettu 28.5.2019. https://github.com/chandlerprall/ThreeCSG/tree/v1

React. 2019. Verkkoaineisto. Luettu 14.4.2019. https://reactjs.org/

Redux. 2019. Verkkoaineisto. Luettu 17.4.2019. https://redux.js.org

Three.js. 2019. Verkkoaineisto. Luettu 2.5.2019. https://threejs.org/